Topics:
- Sorting
- Data Analysis
- Divide and Conquer (Reduction)
- Recursive Algorithm
- Time Analysis

Ex: Shuffling Cards

sort $\{A[1:32]\}$

Reduce 1 problem

sort $(A[1:16])$ $(A[1:32])$ into two of same type

rearranged to $B[1:16]$ such that $B[i] \leq B[i+1]$ and $C[1:16]$ such that $C[i] \leq C[i+1]$

Two sorted sets, how do we put them together?

- compare each card in 1st set to each card in 2nd
- start new sorting problem of size 32
- split into 2 sorting problems of size 16

merge $C[3:16]$

could also do

input $B,C$ such that $B,C$ sorted

output sorted list of $B,C$

divide & conquer: keep reducing problem size till we get to "base" case.

How do we do time analysis?

# of comparisons:

$\text{sort}(32) = \frac{1}{2} \times 32 \times 31 = 512$

$\text{sort}(16) = \frac{1}{2} \times 16 \times 15 = 120$

$\text{sort}(4) = \frac{1}{2} \times 4 \times 3 = 6$

$\text{sort}(2) = \frac{1}{2} \times 2 = 1$

$\Rightarrow 32 + 32 + 32 + 32 + 32 = 32 \times 5$

$\log_2 32 = 5$

reurrence: $n \cdot \frac{a}{n} + \frac{n-1}{2} \cdot \frac{b}{2} = n \lceil \log_2 n \rceil$ for MergeSort
Is this algorithm faster?

Yes, \( \log n \) vs. \( n^2 \)

\[
T_{\log n}(n) = 2T_{\log n}(n/2) + n + n \\
\Rightarrow T_{\log n}(n) = 2T_{\log n}(n/2) + 2n
\]

Another way to think about complexity:

Worst Case Analysis

First way: \( T_{\log n}(n) = T_{\log n}(n/2) + T_{\log n}(n) \)

Recurrence

\[
\Rightarrow 2T_{\log n}(n/2) + 2n = n\log n
\]

Last Time: \( T(n) = T(n-1) + (n-1) \Rightarrow T(n) = n(n-1)/2 \Rightarrow n^2 \)

Solving Recurrences

Suppose \( T(n) = 2T(n/2) + 5n \)

\[
\sum \frac{5n}{2^n} = 5n \cdot \sum 2^{-k} = 5n \cdot \frac{1}{3} = 15n/3 < 5n
\]

Suppose \( T(n) = 2T(n/2) + n^{-\epsilon} \) where \( \epsilon > 0 \)

\[
\epsilon
\]

Another Sorting Algorithm

even a collection of data \( A[1], A[2], \ldots, A[n] \)

What do we want to know? \( \min, \max, \arg, \text{median, mode, std dev} \)

1st Moment \( \sum A[i] \)

2nd Moment \( \sum A[i]^2 \)

3rd Moment \( \sum A[i]^3 \)

Max = 0(n)

Min = 0(n)

Avg = 0(n)

\[
T_{\log n}(n) = ?
\]

\[
T_{\log n}(n) = 2T_{\log n}(n/2) + n + T_{\log n}(n)
\]

\[
T_{\log n}(n) = \Theta(n) \text{ for next time!}
\]